

THE INTERNATIONAL MONTHLY FOR NEXT GENERATION TESTERS

Tea-time with Testers

JUNE 2012 | YEAR 2 | ISSUE V

Jerry Weinberg

The Fish-eye lens

Janet Gregory

Perils and Pitfalls of the 'New' Agile Tester

Bernice Ruhland

Understanding Browser Compatibility Strategies

T Ashok

Do less work - 'Test case immunity' can help

Joel Montvelisky

Use "Cow Magnets" to solve your biggest challenges!

Samarjeet Mohanty

Software Performance Engineering

**If you are reading
THIS**



**then you are one of those
17,900 smart testers
who regularly read**

TEA-TIME WITH TESTERS

Subscribe [here](#) Right Away to get our all Issues for FREE



TEA-TIME WITH TESTERS

First Indian Testing Magazine to reach 97 Countries in the world !

Created and Published by:

Tea-time with Testers.
Hiranandani, Powai,
Mumbai -400076
Maharashtra, India.

Editorial and Advertising Enquiries:

Email: editor@teatimewithtesters.com
Pratik: (+91) 9819013139
Lalit: (+91) 9960556841

This ezine is edited, designed and published by
Tea-time with Testers.

No part of this magazine may be reproduced,
transmitted, distributed or copied without prior written
permission of original authors of respective articles.

Opinions expressed in this ezine do not necessarily
reflect those of the editors of ***Tea-time with Testers.***

Announcing a LIVE webinar

Getting Started with Mobile Application Testing

- 7th July 2012 @ 8:30 a.m. IST -



With constantly changing and challenging world of technology; it's becoming extremely important to test your applications on Mobile Devices. If you are a tester and want to start with Mobile Apps Testing or a fresher who is looking forward to seek career in Mobile Apps Testing, then you must be having multiple questions like:

From where do I start for Mobile Apps Testing?

What path should I follow?

How to seek better career prospects in this field and how can I make a difference here?

We suggest you to join this 90 minutes LIVE webinar (60 mins + Q/A) by **Anurag Khode** and find out how you can "Get Started with Mobile Application Testing".

REGISTER

Editorial

Some new ventures to serve you better!

Dear Readers,

Hope you all are well in health and spirit.

I am pleased to offer you June'12 issue of 'Tea-time with Testers' which is full of wonderful articles as always.

Going forward, along with this treasure of knowledge we will also offer live webinars and online trainings which will help you to enhance your software testing knowledge across different areas.

I am sure that you'll like our additional ventures as we are committed to serve you better every coming time.

We understand the importance and future of 'Mobile Application Testing'. Anurag Khode, who happens to be the founder of 'Mobile QA Zone' and also our core team member, will help you 'Getting started with Mobile Application Testing'. Do not forget to attend his live webinar.

We also have some online trainings on Test Automation which you might find interesting.

Well, now we have offered you almost everything. Articles from legends and experts, live and interactive assistance by SMEs, Puzzles and Crosswords along with prizes.

Now it's your turn to make most of it and become great at testing!

Until next time then!

Sincerely Yours,

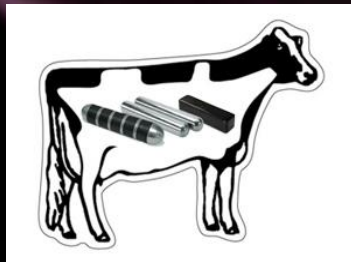


- **Lalitkumar Bhamare**

editor@teatimewithtesters.com



QuickLook



Testing Puzzles
by Sebi



Crossword
by



Editorial

What's making News?

Tea & Testing with Jerry Weinberg

Speaking Tester's Mind

Perils and Pitfalls of New "Agile" Tester – 18

In the School of Testing

Understanding Browser Compatibility Strategies
- 27

Software Performance Engineering – 31

Use "Cow Magnets" to solve your biggest
challenges! -38

T' Talks

Do less work - 'Test case immunity' can help - 44

Testing Puzzle – S.T.O.M. Contest

Our Testimonials

Family de Tea-time with Testers

What's making News?

- find out the latest happenings in the technology world

Deeply-Understanding Static Analysis Testing For Developers

By Adrian Bridgwater, June 14, 2012

Coverity aims to enable developer adoption of static application security testing

Developer testing company Coverity has announced new static analysis technology designed to empower development teams to address security defects in Java web applications.

Combining the firm's static analysis technology and its defect detection tools, the new product aims to extend static analysis to "deeply understand" both source code and modern web application architecture.

The sum result of this so-termed deep understanding is, Coverity says, an opportunity to provide greater accuracy and remediation guidance to help developers find and fix security defects that can lead to the most commonly exploited vulnerabilities including SQL injection and cross-site scripting.

Designed to analyze web applications from the developer's point of view, Coverity's new technology sets out to encourage developer adoption of static application security testing in a way that the company likes to call the "shallow and incomplete analysis" of first-generation tools failed to achieve.

Coverity's tools then augment static source code analysis with a framework analyzer that minimizes inaccuracies when data passes through application frameworks, thereby minimizing false positives. It incorporates a white box fuzzer inside static analysis to automatically validate that data sanitization routines perform sufficient sanitization of un-trusted data and are used in the right context.

"Getting developers to fix security defects requires much more than just integrating static analysis into an IDE. Developers need evidence that the defects identified are real, and they need to understand how to fix those defects in their code," said Andy Chou, Coverity cofounder and chief technology officer. "First-generation static analysis tools are not effective in helping developers because they don't credibly provide them with this information. We are making it easy for developers by taking the guesswork out of finding *and* fixing security defects."

Courtesy- drdobbs.com

For more updates on Software Testing, visit [Quality Testing - Latest Software Testing News!](#)

Are you interested in publishing the news about your own firm, tools, community and conferences in **Tea-time with Testers?**

Then write to us at:

contact@teatimewithtesters.com

with **"News Enquiry"** in your subject line.



*ConditionsApply

Want to connect with right audience?



How would you like to reach over **17,900** test professionals across **97 countries** in the world that read and religiously follow "Tea-time with Testers"?

How about reaching industry thought leaders, intelligent managers and decision makers of organizations?

At "Tea-time with Testers", we're all about making the circle bigger, so get in touch with us to see how you can get in touch with those who matter to you!

ADVERTISE WITH US

To know about our unique offerings and detailed media kit

write to us at sales@teatimewithtesters.com

Announcing...

Smart Tester of the Month Award !!!

❖ Winner for Testing Crossword :

Name: Akash K Patel

Designation: QC Engineer

Company: Silver Touch Technologies Limited

Place: Ahmedabad

Name: Namrata Haridwari

Designation: Jr. QC Engineer

Company: Silver Touch Technologies Limited

Place: Ahmedabad

Name: Sonal Agarwal

Designation: Test Engineer

Company: LogicNEXT

Place: Noida

Congratulations!



Discussion helps !

How about talking with us on Facebook?

Come ! Let's have a nice Tea-time there !

CLICK ON THE PAGE BELOW TO JOIN US



- Wall
- Info
- Photos
- Links
- Discussions
- WELCOME

About

A Free Software Testing Magazine and forum to share, discuss, guide/learn an...

More

202 like this

9 talking about this

Likes

Teach Testing - an

Create a Page

Tea time with Testers ▶ WELCOME

Like

Magazine



"Tea-time with Testers" is a FREE Software Testing Magazine dedicated to all disciplines in the field of Software Testing.

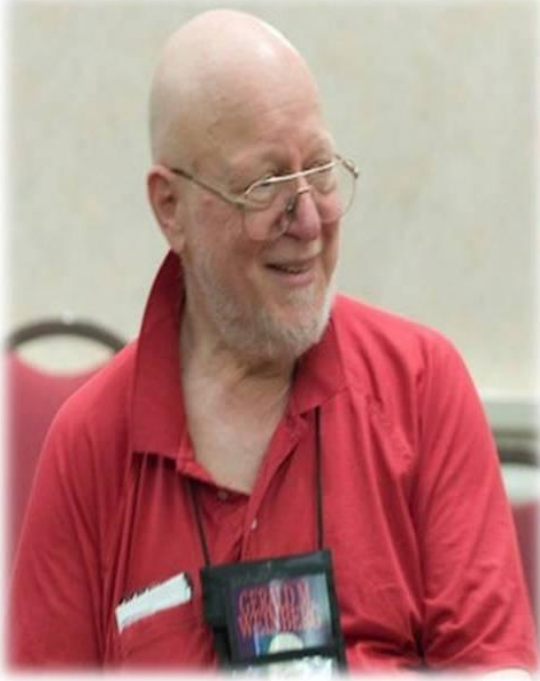
In this magazine you'll get to read variety of articles on software testing & also an opportunity where you can share your own views, win awards, contribute your own ideas, guide others and enjoy every bit of Software Testing.

We are sure , every sip of your tea will be worth enjoying while reading us.



Find us on
Facebook

Tea & Testing



with

Jerry Weinberg

The Fish-Eye Lens (Part 1)

Intricate patterns of effective behavior have grown around the lessons of success and failure, creating a Gordian Knot of Corporate Normalcy ... Every new policy is another hair for the Hairball. Hairs are never taken away, only added. ... The Hairball grows enormous. — Gordon MacKenzie, *Orbiting the Giant Hairball – a Corporate Fool's Guide to Surviving with Grace*. New York: Viking, 1996.

The Fish-Eye Lens represents my ability to see the context—what surrounds me and you, influencing us as we work together. It reminds me to use the many observational and analytical tools I already have, many of which I've written about in my books yet fail to recall when I most need them.

Isabelle's Initial Indication

Remember Isabelle, who volunteered, soon after I arrived at her organization, "We've had consultants before, but none of them made any difference."? I remember her.

In fact, I always have Isabelle on my mind when I enter a new consulting situation, because she reminds me:

You never start with a blank slate.

I call this little reminder Isabelle's Initial Indication. It urges me to retrieve my Fish-Eye-Lens and start looking all around me. All around me.

And why do I need a reminder? Because if I had my druthers, I'd always like to start my consulting with a blank slate. Then I could center and enter and just commence turning willy-nilly, without ever encountering any unanticipated side effects of my interventions. I wouldn't really have to design custom interventions for this client, since they would be blank of history, just like all my other clients would have been but for Isabelle's Initial Indication.

Well, this is a fantasy, sometimes called the Green Field Fantasy, after the idea that it would be easier to build buildings if you always started from nothing, a green field, nice and flat, just growing clover. It's a fantasy because it never, ever happens. That's why I need a Fish-Eye Lens, a tool to tell me why this field, this client, this context, is different from every other client and context I've ever seen before or will ever see again.

The Law of Unavoidably Messy Peculiarity

Well, perhaps each client is not entirely different from every other one. My Fish-Eye Lens reminds me that I have a head-start in determining context because some parts of the context are fixed. These fixed parts are bundled into my lens bag under the title of general systems thinking, a collection of laws that form part of the context of any situation.

For one thing, all my clients (so far) are human beings, and they all have brains.

Moreover, they're all dealing with immensely complex systems, which they must simplify if they are to understand anything at all. In this, they are driven by the Lump Law, a very strong general systems constraint arising from our own mental limitations:

If we want to learn anything, we mustn't try to learn everything.

In computing, certain mathematical transformations are employed to simplify calculation. Underlying these transformations is the idea that certain computations are "stronger," or more difficult, than others—that is, they require more computational power, or more time to compute. By transforming a problem, we may reduce the "strength" necessary to solve it. And, indeed, my clients do something similar: They don't deal with raw data about their environment; they use transformed—simplified—data.

That simplification helps them function successfully, but it may not help me obtain the data behind their transformations. Because we each lump the world in our own unique way, One person's help is another's hardship.

I call this the Law of Unavoidably Messy Peculiarity (LUMP, because it's a corollary of the Lump Law). Consultants are also limited by the Lump Law, but there's a way around it—one of the true secrets of consulting. To explain, I have to tell a story about my youth that will reveal more than I would like about just how ancient I am, but—oh well.

First Law of Good Consulting

When I was in college, there were no electronic computers anywhere on campus—indeed, there were just three or four in the world. But there were computers—and in fact, I was one. I was paid 90 cents an hour by the physics department to perform long, laborious, and tedious calculations, using pencil, paper, and a huge, loud mechanical Friden calculator. The Friden helped because it could do large

multiplications in a several seconds of clattering and clunking, rather than the minutes it would have taken me to do it with pencil and paper. Still, it took long enough that whenever I had to compute a formula with something like

$$B = 3 * A$$

I would translate it to a weaker form,

$$B = A + A + A$$

for even though there are now two additions instead of one multiplication, the additions are weaker—that is, I could do these two additions faster than the Friden could do one multiplication.

Similarly, the formula,

$$B = A^3 \text{ (A to the third power)}$$

might be weakened to

$$B = A * A * A$$

because two multiplications were much easier (faster) than a single exponentiation, which, in fact, neither the Friden nor I could do at all without a table of logarithms.

This is the idea underlying logarithms: a single multiplication is "reduced" to the following steps:

- look up logarithm of A
- look up logarithm of B
- add the two logarithms
- look up the antilogarithm

When I was in high school, we were taught this rather formidable process because we had no calculators, not even Fridens. Fridens made multiplication almost as "weak" as addition, and nowadays, the environment has changed so much that calculators have made exponentiation as weak as multiplication and addition. Nobody teaches this particular transformation, because even if I need exponentiation, I don't need logarithms

—as long as I have one of those free calculators that are given away by the corner gas station. Today's young whippersnappers don't realize how lucky they are! Even in my most maudlin nostalgic moments, I don't miss logarithms one bit.

Now that we have electronic calculators and computers, we don't need these particular strength-reducing transformations, but the human mind—my mind, certainly—remains rather fixed in its reasoning abilities.

For some of us, pictures are much easier to comprehend than words, so transforming a set of numbers into a graph is a strength reducing operation. Others, however, have never grasped graphs, but easily understand stories that give essentially the same information. Or the original numbers. Or a metaphor, or picture, or formula. What's weak for some is strong for others.

When I'm getting information from lots of different people—as I usually do to establish context—I have to be equipped with a variety of potentially strength-reducing transformations, including,

- words to pictures of various kinds
- pictures to words
- numbers to metaphors
- metaphors to pictures
- graphs to stories
- stories to numbers
- words to three-dimensional physical models
- words to formulas
- concepts to examples
- examples to concepts
- formulas to pictures
- formulas to numbers
- numbers to pictures
- numbers to words
- words to actions, or demonstrations
- actions to pictures
- actions to formulas
- and many, many others.

These transformations are all part of my Fish-Eye Lens. I think of them as different filters. I'm prepared to use whatever filter—to make whatever strength-reducing translation—allows all clients to use their easiest mental forms. And, I also come prepared to test for what's easiest by fitting my Fish-Eye lens with one filter, then another.

Remember the Law of the Hammer?

The child who receives a hammer for Christmas will discover that everything needs pounding.

Any photographer can tell you that hammering is not likely to improve a filter, yet sometimes when I'm frustrated trying to understand a client, I start hammering away using one and only one of my filters. This is the First Law of Bad Management, or, if you like, the First Law of Bad Consulting: If something isn't working, do more of it.

My Fish-Eye Lens, with its case full of filters, helps me remember not to be a bad consultant—at least this kind of bad consultant.

to be continued in next issue...

Back To Index



Are you designing training for Testers?

Well, then you must read Jerry's one of the latest book. It is...

Experiential Learning: Beginning

Gerald M Weinberg

"Telling is not Teaching"

Read the preface of this book [here](#).

Buy the ebook now!

When you buy this book, you get it in PDF, EPUB and MOBI formats, so you can read it on your computer, iPad, Kindle or other ebook reader!

If you buy the book, you get Jerry's all the Leanpub updates to the book for free!

Biography

Gerald Marvin (Jerry) Weinberg is an American computer scientist, author and teacher of the psychology and anthropology of computer software development.



For more than 50 years, he has worked on transforming software organizations. He is author or co-author of many articles and books, including *The Psychology of Computer Programming*. His books cover all phases of the software life-cycle. They include *Exploring Requirements*, *Rethinking Systems Analysis and Design*, *The Handbook of Walkthroughs*, *Design*.

In 1993 he was the Winner of the **J.-D. Warnier Prize for Excellence** in Information Sciences, the 2000 Winner of **The Stevens Award** for Contributions to Software Engineering, and the 2010 **Software Test Professionals first annual Luminary Award**.

To know more about Gerald and his work, please visit his Official Website [here](#).

Gerald can be reached at hardpretzel@earthlink.net or on twitter @JerryWeinberg

More Secrets of Consulting is another book by Jerry after his world famous book **Secrets of Consulting**.

This book throws light on many aspects, ways and tools that consultant needs.

“Ultimately, what you will discover as you read this book is that the tools to use are an exceptionally well tuned common sense, a focus on street smarts, a little bit of technical knowledge, and a whole lot of discernment”, says **Michael Larsen**.

More Secrets is definitely useful not only to consultants but to anyone for building up his/her own character by implementation of the tools mentioned in day to day life.

Its sample can be read online [here](#).

To know more about Jerry's writing on software please click [here](#).

MORE SECRETS OF CONSULTING



Gerald M. Weinberg

TTWT Rating: ★★★★★

A photograph of a green, conical pendulum bob hanging from a thin wire. The bob is positioned over a light-colored sand surface. In the sand, there is a faint, circular mandala-like pattern. The text "Speaking Tester's Mind" is overlaid on the image in a large, blue, serif font with a white outline.

Speaking Tester's Mind

- straight from the author's desk

Perils & Pitfalls of the New “Agile” Tester

-Part 2

by Janet Gregory

5. Maintaining a “Quality Police” Mindset

On agile teams, we want the team to determine when it is ready to ship. The team includes testers, programmers, business users, etc. However, the final decision on any project is a business decision based on business needs and risks.

In traditional projects, testers often have the best understanding of the quality of the product since they see it as a whole and report the defects. When I was a QA Manager on projects like these, it was my job to say whether I thought it was ready or not, and I took the tester’s opinion very seriously. It is hard to let go of the quality police mindset.

When you are transitioning to agile for first time, it may be difficult to let go of the separate test team. You may want to insist that all bugs are recorded in the defect tracking system for quality metrics, or the test team has the right to ‘Stop the release’.

The risk of falling into this trap is that rest of the project team lets the testers maintain the control, and never actually buy into the “build quality in” concept. Testers become the safety net for the programmers.

The avoidance of this peril is difficult because it requires a complete change of mindset and a belief in the new system.

Some of the things you can try are:

- Define “Done” up front so that testing is defined as part of each story
- Try cards on the story/task board for bugs found on stories
- Talk the talk - get the whole team to own the “quality” of the product
- Show how each role adds value and quality
- Develop a good relationship with programmers
- Share your expertise, rather than hoarding it
- Ensure that testing happens during the iteration
- Bring up testing issues during the iteration retrospective so the whole team has input and can come up with team solutions

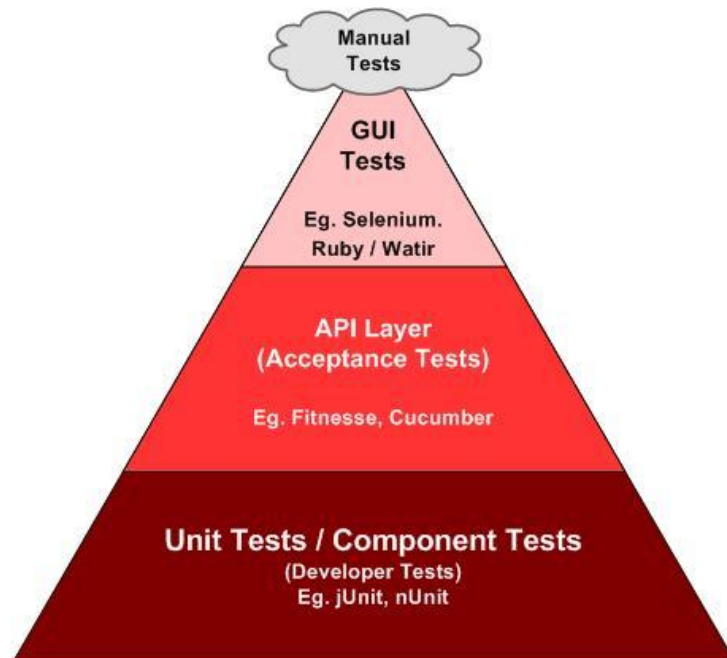
6. Trying to Test Everything Manually

Many organizations still try to test everything manually. There is nothing wrong with manual tests, and in fact they are very useful for exploratory testing or usability testing. However, manual regression testing is repetitive, boring and error prone.

If you seem to spend all your time retesting features already tested, and not getting to new features, you have fallen into this trap. Your team will struggle just to keep up.

The risk is that bugs that are already fixed creep back into the code because the application cannot be completely tested each time when there is a new build or a new feature. Testing cannot keep up with the new stories and are only partially tested. Features that used to work and are now broken aren't noticed. Constantly changing code base makes manual test maintenance burdensome.

Automation is the key. There are several types of automation that can make up your regression suite. Using the “automated testing pyramid” model, unit tests created by the programmers are the base that allows refactoring in safety. The next layer is the functional tests created from your acceptance and story tests at the API level. The top of the pyramid is reserved for the automated tests through the GUI.



There is overhead involved but the return on investments is worth it if the tests are well written.

- Automated suites become your change detector
- You can help programmers to write good unit tests
- Work with programmers to automate functional tests – they are experts in developing
- Find an automation tool that works for the entire team
- Automate as much as you can
- Include time in your estimates for test automation and maintenance
- Design for testability (because whole team involved in testing)

7. Thinking You Are Not Technical Enough

Much of the literature about agile says that it is helpful to have automation experience, therefore many testers think they can't work on an agile project. However, an agile team requires many different talents and there is room for everyone. If there was someone on the team who was not good at programming or testing, but was really good at identifying important concepts and describing them clearly, that would be a bonus. Whatever the distribution of skills and abilities on the team, if they're functioning well as a team, people will accommodate that distribution to get the job done.

You will recognize that you have fallen into this trap if you are hiding during iteration meetings or not contributing to implementation or design discussions.

The risk to the team is that they don't get the benefit of tester's skill set. There is no interaction with the programmers so testers don't learn anything new and progress.

To avoid this trap, or help you get out of it,

- Don't be afraid to ask questions
- Show your skills – testing skills are technical
- Communicate, communicate, communicate
- Ask for help
- Contribute in areas you know to start
- Self-study using on-line webinars, blogs, articles, books, even courses
- Expand your toolkit and become technically aware

8. Forgetting the Big Picture

Iterations are comprised of stories that are just small chunks of functionality. It is easy to fall into the trap of testing each of these stories and forgetting that they are part of a feature or bigger product.

You know you have forgotten the big picture if you are finding integration bugs late in the release, or the workflows that don't make sense, or reports aren't getting developed until the end because you forgot about them.

The risk is that stories don't connect and thus gaps are created. Customer won't accept the product because of usability issue, or there are missing pieces of functionality.

Keep the end goal in mind even if you make adjustments along the way.

Some tips to help you look at the big picture are:

- Think about workflows when you are dividing the feature into stories
- Think about impacts to other parts of the system
- Find way to build test data that reflects 'real world'
- Use whiteboards to draw pictures
- Use business-facing tests to help drive development
- Use exploratory testing to see how stories fit together

Conclusion

Being a new agile tester, and armed with the understanding of some of the pitfalls that wait for you, does not mean you will recognize the problems or be able to effect change in your organization.

I strongly recommend having an experienced agile for new teams so they can help to identify some of the issues and ways to overcome them and make changes in the team.

References

- Lisa Crispin and Janet Gregory, *Agile Testing: A Practical Guide for Testers and Agile Teams*, 2009, Addison-Wesley
- <http://www.testobsessed.com> - Elisabeth Hendrickson's web site
- Gojko Adzic, *Bridging the Communication Gap*, 2009; *Specification by Example*, 2011
- <http://lisa.crispin.com> - Lisa Crispin's web site and blog
- janetgregory.blogspot.com - Janet's blog
- Agile Manifesto: <http://agilemanifesto.org/>
- Linda Rising and Mary Lynn Manns. *Fearless Change: Patterns for introducing new ideas*
- Yahoo Agile Testing group agile-testing@yahooglegroups.com
- Lisa Crispin and Tip House, *Testing Extreme Programming*, 2002, Addison-Wesley

Back To Index



An agile testing coach and practitioner, Janet Gregory is the co-author of *Agile Testing: A Practical Guide for Testers and Agile Teams* and a contributor to *97 Things Every Programmer Should Know*. Janet specializes in showing agile teams how testers can add value in areas beyond critiquing the product; for example, guiding development with business-facing tests. For the past ten years, Janet has been working with teams to transition to agile development, and teaches agile testing courses and tutorials worldwide. Janet contributes articles to publications such as *Better Software*, *Software Test & Performance Magazine* and *Agile Journal*, and enjoys sharing her experiences at conferences and user group meetings around the world. Janet was named one of the 13 Women of Influence in testing by *Software Test & Performance* magazine – January 2010.

For more about Janet's work, visit www.janetgregory.ca or visit her blog at janetgregory.blogspot.com/





Do **YOU** have **IT** in you what it takes to be **GOOD** Testing Coach?

We are looking for skilled **ONLINE TRAINERS** for Manual Testing, Database Testing and Automation Tools like Selenium, QTP, Loadrunner, Quality Center, JMeter and SoapUI.

TEA-TIME WITH TESTERS in association with **QUALITY LEARNING** is offering you this unique opportunity.

If you think that **YOU** are the **PLAYER** then send your profiles to trainers@qualitylearning.in .

Click [here](#) to know more

There was a time when people did not have compass to find right direction. The only guide they had was that guiding star up in the sky.

Do you think that you are also stuck somewhere with technical issues? Do you need help in decision making or want guidance?

Well, the wait is now over . Introducing...

“The Guiding Star”

*The panel of our experts is now here to help you.
Send us your questions around software testing and our Guiding Stars will help you out.*



E-mail your question on –

theguidingstar@teatimewithtesters.com

Please Note :

1. This is not a job portal.
2. Typical interview questions will not be answered.
3. Questions should be on Software Testing or related topics only.

Do you have any Questions or Feedback on articles that we publish in Tea-time with Testers?

No Problemo! We will publish your Feedback/Comments and also the answers to your Questions that you have for our Authors.

Do write us your Feedback and Questions in below format and send it to teatimewithtesters@gmail.com :

- Your Name
- Your Brief Introduction
- Article Name
- Your Feedback or Questions if any

➤ Your Feedback or Question

Make sure to write **Feedback For < Article Name>** in your subject line.



A photograph of several students in a classroom, seen from behind, with their hands raised in the air. They are facing a chalkboard that has some faint writing on it. The students are wearing colorful shirts: light blue, red, orange, and green. The entire image is framed by a thick black border.

In the school of Testing

for your better learning & sharing experience

Announcing FREE Live webinar

“Getting started with Automation Testing”

~ 4th July 2012 @ 9PM IST ~

Register [HERE](#) right away !

Live and Interactive **"Quick Test Professional (QTP)"**

Online Training batch starts on

6th July 2012 @ 7AM IST

Click [HERE](#) to Register !

"Selenium Basic and Advanced" Online Training batch
starts on 13th July 2012 @ 8AM IST

Click [HERE](#) to Register !

Understanding Browser Compatibility Strategies



By Bernice Niel Ruhland

The Importance of Social Networking – Part 1



Since November 2011, I have published a series of articles on how testers can develop their skills and knowledge. I would like to take a different approach to discuss how I apply my own advice. Recently, I had to learn how to effectively test a web-based application across multiple browsers. To accomplish this task I needed to understand how other testers approach this problem and the tools they are using. I am fortunate to have a large testing network through social media and that many of these testers are bloggers. In addition, I have interacted with many of them through different forums to discuss testing challenges and approaches.

The focus of this series does not discuss how I perform cross-browser testing. Instead it provides a wealth of information shared from the Testing Community and information from my personal research. Review the suggestions to identify the strategy that best meets your browser compatibility goals.

I would like to thank everyone who contributed information. Without their willingness to share their experiences and recommendations, I would not have been able to understand potential approaches in such a short time. A heart-felt thanks to: Ajay Balamurugadas, Mike Talks, Lisa Crispin, Martijn de Vrieze, Anne-Marie Charrett, Karen Johnson, Gagneet Singh, Dave McNulla, Moise Stedte, Akshay Thakkar, and Dorothy Graham.

Define Goals and Problems

My initial step was to understand my goal and what problems/questions that required answers. My overall goal was to test a web-based application across multiple browsers. My questions included:

- How do you get started with understanding differences between browsers?
- How do you test across browsers and versions?
- How do I know how much time to allocate to testing other browsers?
- What tools are available to assist in testing?

Conduct Research

Once I had my goal and questions defined, I started to conduct the research. I posted questions on Twitter, G+, and I emailed testers that might have experience in this area. The responses were shared via G+, email, spreadsheets, and Skype. Plus I read any blogs and articles on performing cross-browser testing and tools. I compiled the responses in a Word document categorizing the responses to the appropriate questions. In addition, many people provided information that went beyond my initial questions.

Testers may have different strategies but there were a lot of themes or similarities on how testers are approaching this problem.

How do you get started with understanding differences between browsers?

The most common approach is to manually test your functionality across multiple browsers to understand the differences. Or at least that is the common perception. Some testers are using a tool that generates screen captures across browsers allowing them to manually blink test for differences. This also allows you to capture problems in design and other types of bugs.

A good blog on using a tool to get started with browser testing: Eliminate boring testing: Automating Visual Comparison <http://trishkhoo.com/2012/04/eliminate-boring-testing-automating-visual-comparison/>

I enjoyed reading Mike Talks blog on his actual experience in testing across browsers and recommend it: Rapid Exploratory Website Testing. <http://testsheepnz.blogspot.co.nz/2012/04/rapid-exploratory-website-testing.html>

How do you test across browsers and versions?

There are a variety of approaches to determine how to test across browsers.

Browsers Test Strategies

- Identify the most popular browser used by customers to perform functional testing. Once functional testing has been completed, transition to Compatibility Testing. For Compatibility Testing, identify the minimal subset of testing that must be performed based upon browser risks. Determine usage of browsers/versions by customers or market share to identify which browsers and versions are part of the Compatibility Testing. Try to identify usage statistics to help support decisions to upper management.
- What percentage of your clients is your company looking to support? 80%? 90%? Understanding this business decision can determine how many browsers you need to test.
- Do not try to test all browsers and versions because the time invested versus bugs identified will be a small return. Consider the likelihood of bugs being fixed for browsers with low usage. If the bug will not be fixed, then do not test it.
- Determine strategies to cycle different browser versions into testing using several testers. For example, a spreadsheet can be used to identify testing features and browsers for each tester to perform. If a module has four testing parts, assign different parts to be tested against a specific browser and version. Rotate the testing so over time the module is tested across all supported browsers and versions.
- Perform Compatibility Testing with a defined group of browsers and versions. During regression testing, identify strategies to rotate other browsers or versions into testing.
- Take the allocated time for testing and divide it across the browsers to test the functionality. If the tester has 8-hours to test and 4 browsers, spend 2 hours on each browser.
- Consider using a tool such as HEXAWISE to help identify testing coverage. For more information see their website: hexawise.com
- Understand what browsers the developers are using as part of their testing. Testers could use other browsers to allow more testing coverage.
- As newer versions of browsers are added, consider allocating more testing time based upon potential usage by customers. A reduction in testing of other browsers will be

determined at that time. It is important to consider when to sunset browser versions when new versions are added.

- If you need to test on multiple browsers and versions, select one version of each browser to test. If there is a high risk that the new code will fail in other versions, consider testing on those versions for that functionality.
- Do not make general claims of browser independence, as it is better to define specific browsers and versions supported.

Adding and Sunsetting Browser Versions

- Develop a plan to determine when to add and sunset browser versions. This includes understanding usage statistics of different browsers coupled with information on what browsers your clients are using.
- It is important to review usage statistics of browsers to understand how it changes over time. At one time IE had the largest market share. Over the years, Chrome and Firefox are gaining popularity challenging IE as the market leader. There are many sources to identify percentage usage of browsers. One source to use: http://en.wikipedia.org/wiki/Usage_share_of_web_browsers
- Periodically review what browsers and versions your clients are using or would like to use with your product. How is this information different than what is currently supported? Do you need to make any changes?

Back To Index



Bernice Niel Ruhland is a Software Testing Manager for ValueCentric, LLC a software development company located in Orchard Park, New York. She has more than 20-years experience in testing strategies and execution; performing data validation; and financial programming.

To complete her Masters in Strategic Leadership, she conducted a peer-review research project on career development and on-boarding strategies. She uses social media to connect with other testers to learn more about their testing approaches to challenge her own testing skills.

The opinions of this article are her own and not reflective of the company she is employed with. If you have any questions / comments on this article or if you would like to connect, Bernice can be reached at:

LinkedIn: <http://www.linkedin.com/in/bernicenielruhland>

Twitter: [bruhland2000](#)

G+ and Facebook: Bernice Niel Ruhland



Software Performance Engineering :



‘So many types of tests – Confused !!’

by Samajeet Mohanty

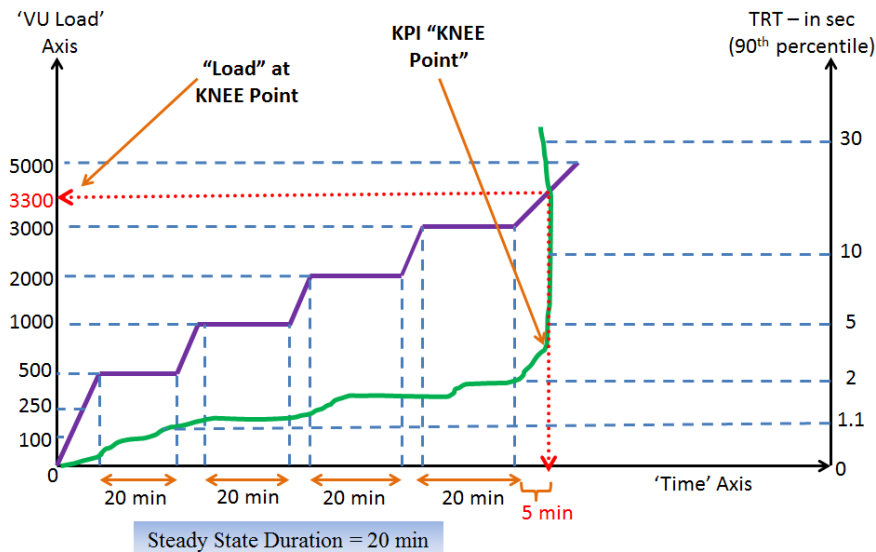
(continued from April'12 issue)

Scalability/Capacity Test:

GOAL: Find out if and how the application scales when subjected to incremental load beyond maximum expected user load.

WHAT DOES THIS MEAN ?

Keep the environment tunables configured as is and simulate a load starting from 500 virtual users incrementing in small steps till a Knee Point is reached for any of the performance KPIs like TRT, Throughput, application error count and so on.



Things to Note:

1. The load increment factor & steady state duration depends on the balance of how quick you want to identify Knee Threshold during the test.
2. In Figure above, Response Time KPI is measured against the load increment.
3. Till a VU load of 3000 simultaneous users, response time was well under ~2 sec. However within 5 min of load increment when user count reached to 3300, response time spiked upwards indicating massive performance degradation. This is your Knee Point as depicted. You would probably notice application related errors in log files or a heavy resource utilization at the OS level which might have caused this.

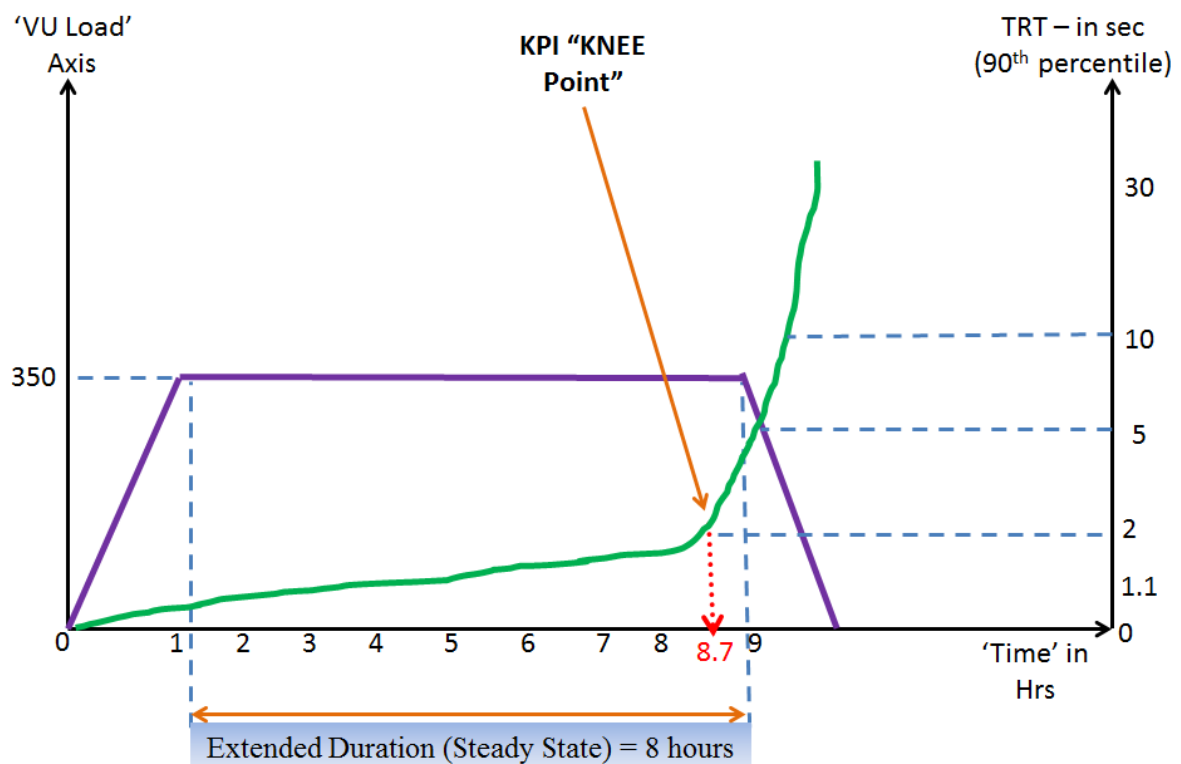
Scalability test, in a way, tells us the load handling capacity of current infrastructure. The load can be of any form – online transaction processing or back-end message processing or some other mechanism which has the ability to inject 2 times or even 3 times expected load and verify at what point current-setup breaks. This kind of test helps the architect to plan better for future hardware (HS), software upgrades and application technology changes for increased capacity requirements.

Endurance Test: (aka: Reliability / Durability Test)

GOAL: Find out application behavior when subjected to a continuous load for extended duration of time.

WHAT DOES THIS MEAN ?

Keep the environment tunables configured as is and simulate an average virtual load doing day-to-day operations, for say 1 business day (8 hours). The test duration can be extended to 24 - 48 hours too depending on geographical access for the application.



Things to Note:

1. Reliability Test is done with an average load condition but not peak load (500 in our case). In reality an application experiences high usage spikes only intermittently during a day, so it would be safe to assume that system is constantly under load only by an average of expected peak load scenario.
2. As you can see, the response time remained within threshold till the 7th hour. However before completing 8 hours of test, the application performance degraded sharply. This could have happened due to many underlying reasons like process count, thread count, data pool size exhaustion and so on. To identify the root cause of issues, these test results need to be correlated with other monitored statistics like CPU/Memory/IO usage.

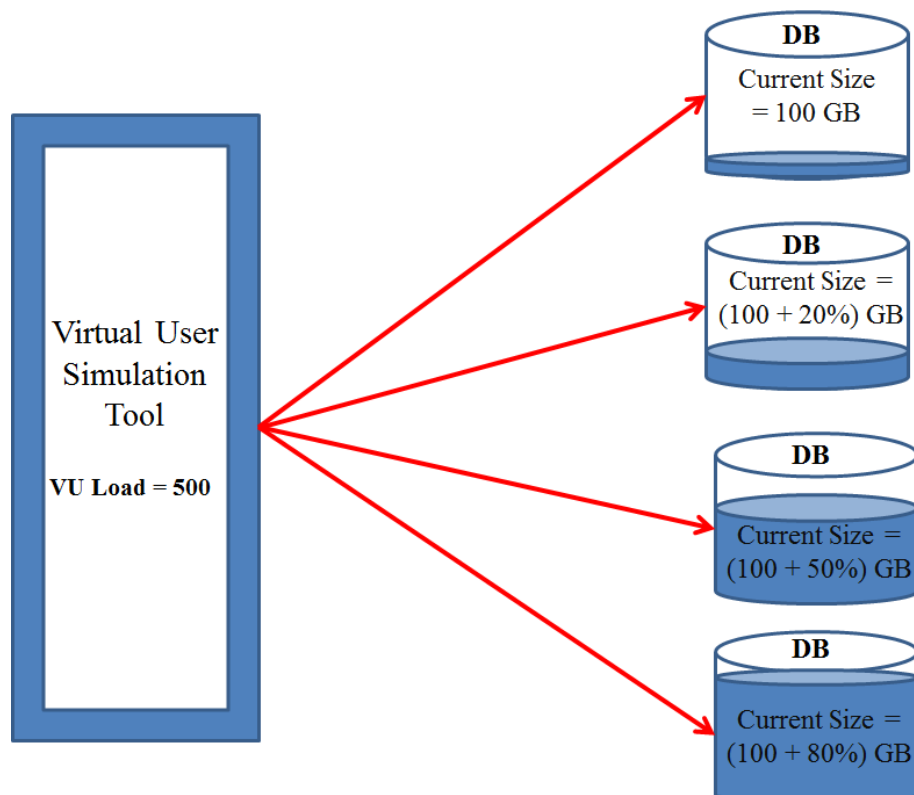
In Reliability test, SUT should be subjected to a load continuously for an extended period of time to verify the kind of effect longer working conditions have on an application resources and on the hardware it's hosted on.

Volume Test:

GOAL: Find out application's behaviour when it is subjected to user load with variable application data volume in database.

WHAT DOES THIS MEAN ?

Vary the data volume in database by ramping up from current till expected and then 3 times the expected DB volume. The maximum user load would be simulated against these varied DB data volumes.



Things to Note:

1. By Volume Test here, I'm referring to the database growth.
2. With the application running from multiple geographies 7*24, it is imperative that the application data volume would grow significantly over time. So, the load tests conducted prior to the launch application with a set DB volume needs to be re-done with revised set of data volume growth expectations.
3. As you can see from the depiction above, the maximum expected user load (of 500) is simulated against incremental growth sizes to current database size. The increment factor depends on how fast you expect the DB volume to bulk up.

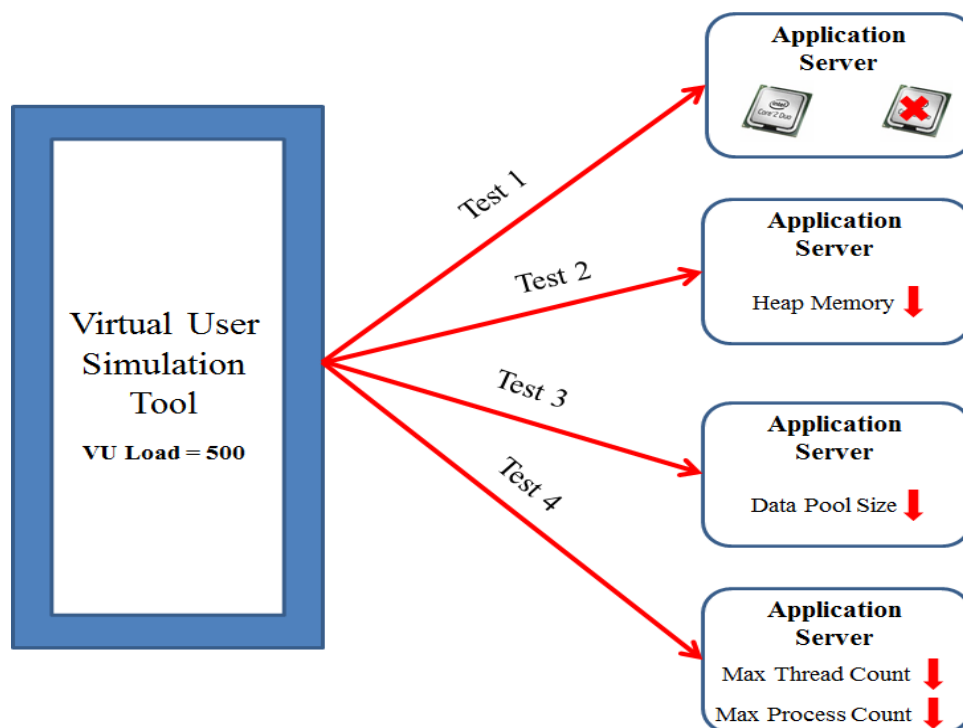
Volume Test is generally conducted for applications that use data warehouse like technologies and consist of transactions involving heavy input data volumes.

Stress Test:

GOAL: Find out if and how an application handles the unexpected stress put on it during normal working conditions.

WHAT DOES THIS MEAN ?

Introduce such an environment for SUT that it finds stressful to work under and may face under normal working conditions. The 'how' part is verified by looking at the type of alerts or errors reported by environment / application under stress. There can be two categories of stress: *Positive* and *Negative*.



Things to Note:

There are lots of ways to put a SUT under stress. Some of them include, when operating under normal conditions (with system loaded):

1. Shutdown 1 CPU
2. Lower allotted heap memory to application from max 1 GB to say only 512 MB
3. Lower the data pool size to say half of original size
4. Lower the max thread or process count running in parallel

Above conditions should be introduced when the system is full-on active and all infrastructure resources are being utilized. Please note that above depiction talks about only application server. Database server can be subjected to same kind of resources constraints.

Bottom-line, cut down on resources for any of the infrastructure component and verify how the SUT behaves. This is a Negative Stress scenario or Robustness Testing. Scalability Test on the other hand can be considered as a Positive Stress scenario.

Component Test:

GOAL: Find out if an infrastructure component lives upto its performance expectations.

WHAT DOES THIS MEAN ?

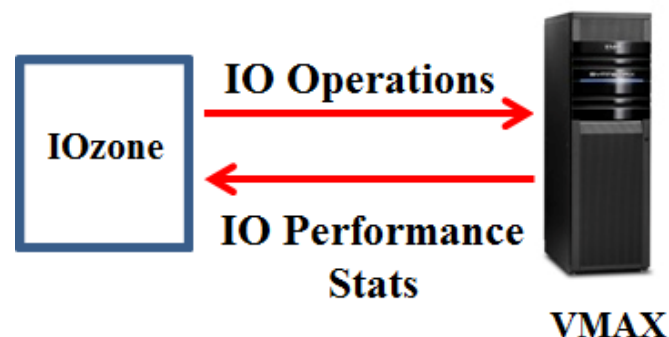
A component test is any performance test that targets an architectural component of an application. Commonly tested components include servers, databases, networks, firewalls, clients, and storage devices [1].

Example:

I have had the opportunity to be part of a team that is virtualizing its data center and for this they are going for VCE Vblock. One of the requirements of project team was to test the performance of Vblock's storage component "Symmetrix VMAX" and have it compared against vendor's SLA.

Talk about not taking vendor's word for granted.....huh !!.

Anyways though not directly involved with this test, I did get a chance to hear the architect who was performing it. After lot of R&D to find the appropriate mechanism for this kinda testing, he finalized on IOzone[2] which is a file system benchmark tool.



As seen in Figure above, IOzone issues different types of IO operations like read, write, parallel read & write so on against the disk array of VMAX and calculates the IO throughput (IOPS) based on the results obtained from the storage component. Few other tools to test storage component performance are Bonnie++, Orion and Iometer.

In conclusion, irrespective of the type of test you want to do and call it whatever you want, be sure to accurately reflect PROD activity like OLTP transactions happening in conjunction with any CRON jobs scheduled to run the EOD processing and stuff like that. The last thing you want is do a test and find out later that it does not simulate the expected production behaviour.

REFERENCES:

- [1] <http://msdn.microsoft.com/en-us/library/bb924357.aspx>
- [2] <http://www.iozone.org/>

APPENDIX:

KPI - Key Performance Indicator

EOD - End-of-Day

FTP - File Transfer Protocol

SGA - System Global Area / Shared Global Area

VU - Virtual User/s

TRT - Transaction Response Time

HS - Hardware Sizing

90th Percentile - This KPI applies to TRT. It stands for response time value of 90% of VU load applied.

SUT - System under test

OLTP - Online Transaction Processing



Samarjeet Mohanty (Samar) is a, self-proclaimed, practitioner of anything to do with Software Performance Engineering world.

He's quite experienced in Performance Engineering and Testing methodologies of software applications (BFSI) using industry standard tools and techniques.

Apart from being an active participant in technical forum's concerning performance and generic testing QA, Samar likes to interact with creative-minded professionals from all walks of life to better understand their take in the related field.

If interested, connect on:

LinkedIn:

<http://ca.linkedin.com/in/samarjeetm>

Twitter:

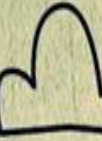
<http://twitter.com/SamarjeetM>

Back To Index





are you one of those
#smart testers who
know d taste of #real
testing magazine...?



then you must be telling your friends about ..



Tea-time with Testers

Don't you ? 😊



Tea-time with Testers !

first choice of every #smart tester !



testing intelligence

- its all about becoming an intelligent tester



an exclusive series by **Joel Montvelisky**

Use “Cow Magnets” to solve your biggest challenges!



My son & our guide milking a cow

This weekend I went with my family to a Robotic Dairy Farm.

Obviously the kids had a blast with the cows, and my oldest son made me proud by showing off his cow-milking skills, or at least how he's not afraid to try 😊

How is this related to testing???

Let's start by saying that after having done both these jobs during my professional lifetime, milking a cow is nothing like testing software.

Still during the visit we learned something that reminded me of how we should always look for alternative (and sometimes unconventional) approaches to solve our biggest testing challenges.

Solving the “Hardware Disease” in the Dairy Industry

Did you know that there is such a thing as “**Hardware Disease**” that harms dairy cattle? (and NO!, it is not related to a cow using a mouse and hurting its wrist in any way!!!).

Based on what they explained to us in the farm, up to some years ago close to 1% of all dairy cows would die as a result of eating metal objects (together with the hay) that would get in their stomachs, perforate them and cause large internal bleeding.

To solve this issue they first tried to develop technology that would “find and extract” all metal objects from the hay before feeding it to cows, but this approach proved both expensive and impractical.



Then, someone thought about an unconventional approach: “Instead of fetching them before they are eaten, let’s prevent the cows from dying even if they ingest these metal objects”. And they came up with the idea of feeding cows a relatively small but powerful **Cow Magnet** that would sit on their “second stomach” and catch metal objects before they would cause any harm to the animal.

As weird as it sounds this solution worked, and it is now widely used in the Dairy Industry saving thousands of cows a year world wide.

Looking for our own “Cow Magnet” solutions

Now let’s get back to testing...

How many times have you gotten stuck trying to solve a problem using conventional approaches instead of trying to think “outside the box” looking for unconventional wisdom?

When this happens it is usually because you are focusing on the wrong problem, like in the case of the cows where they tried to tackle the issues of the metal objects instead of focusing on the death of the cattle.

Let’s use an example of a testing challenge and how we solved it to explain what I mean.

A “Cow Magnet” to solve DB migration failures

Long before I started working in PractiTest, I was managing the QA for another enterprise software company. The product I was in charge of testing ran locally on our customer’s servers (this was before the world learned about the **advantages of SaaS** software!), and we had thousands of installations world-wide, ranging from a handful of licenses and all the way to customers with thousands of end-users.

We had a problem in this company. Each release (once or twice a year) we had to migrate the database to a new schema, and for a large number of the customers (who had made modifications mainly to create their own reports) this meant that their “version upgrade” would get stuck during the db migration phase, and they would need to get someone from our support to manually “fix” the issues and complete the migration process.

This resulted in large number of organizations that were afraid to upgrade, and our product quickly started getting a bad reputation in the field.

In the beginning we defined the testing problem as “been able to generate all these db customizations internally in the lab”, but after 2 releases of trying this approach the results in the field were still disastrous.

Then we started searching for our “Cow Magnet”, or in other words we re-defined our challenge this time to “making sure as many of the problematic databases would be able to migrate successfully”. Notice that instead of focusing on the “testing artifact” we focused on the problem from the perspective of our users, and this was the big breakthrough.

We got in touch with our support team and asked them to contact all customers who had experience migration issues in the past and ask them if they would provide us with a copy of their databases.

We got our hands on close to 40 such projects (only about 1/3 of the organizations agreed to send us their DBs) and we created an automatic framework that would run the upgrade on all of them once a week verifying the results of this operation (a simple pass or fail test), and informing Development whenever one of their changes “broke” the migration.

By running these tests we found that each release had only between 10 to 20 operations that would be responsible for most of the migration issues. And within one release we reduced the upgrade issues from around 6% to less than 0.3% of all upgraded projects.

Three quick methods to find your “Cow Magnets”?

Looking for your “Cow Magnet” is not hard, and the main challenge is to re-define your problems in a way that will let you find “other” possible solutions.

Following are three methods I use to achieve this.

Method 1 – Look at the problem from your user’s perspective

This is the simplest method and one you can even do by yourself, simply try to put yourself in your customer’s shoes and think how he sees the problem.

Like in the example above, the user didn’t care the problem was caused by his customizations, he simply wanted his project to be migrated successfully. Or in the case of the dairy farmers, they didn’t care about the pieces of metal, they simply wanted their cows not to die.

Method 2 – 5 Whys

To quote [Wikipedia](#) on it: “The **5 Whys** is a question-asking technique used to explore the cause-and-effect relationships underlying a particular problem. The primary goal of the technique is to determine the root cause of a defect or problem...”

Basically, take some of your peers and write down the problem you are trying to solve on a whiteboard, then ask WHY? (e.g. why are looking to take out the metal objects of the hay, or why are databases failing to upgrade?).

Write the answer/s you got, and ask once again the question WHY?

Repeat this process 4 to 5 times and you will get to the source of the issue you are trying to solve and hopefully to more ways how to solve it.

Method 3 – Brainstorm with other members of your Organizations, but not from your own team!

This might be the most effective method since it will give you the broader results, but it is also the most time consuming, so use it only when the problem you are looking to solve is important enough.

Set up a meeting where you bring people from other teams in your company such as Support, Sales, Marketing, Finance, HR, etc. Then perform a brainstorming session with them around your problem (you can use the 5 whys, or any other **brain-storming** method you like) and try to bring as many ideas as possible into the table.

The advantage of this method is that it will provide a lot of varied approaches that are not biased by the technical nature of the testing and development teams.



Have you come up any “Cow Magnets” in the past? Share them!!

An additional example that comes to mind is the tale about the Russian Space Pencil: As part of the Space Race it became apparent that pens (that work based on the principle of gravity) don't work in outer space. The Americans spent millions of dollars developing a pen that would work on Zero-G conditions (and also under water, at extreme conditions or cold and heat, etc), while the Russians simply gave their cosmonauts pencils... (BTW, this is apparently only an urban legend but still drives the point home)

Have you come up with “Cow Magnets” of your own?

Share them with us and give additional ideas on how to go around solving problems using unconventional approaches!



Joel Montvelisky is a tester and test manager with over 14 years of experience in the field.

He's worked in companies ranging from small Internet Start-Ups and all the way to large multinational corporations, including Mercury Interactive (currently HP Software) where he managed the QA for TestDirector/Quality Center, QTP, WinRunner, and additional products in the Testing Area.

Today Joel is the Solution and Methodology Architect at PractiTest, a new Lightweight Enterprise Test Management Platform.

He also imparts short training and consulting sessions, and is one of the chief editors of ThinkTesting - a Hebrew Testing Magazine.

Joel publishes a blog under - <http://qablog.practitest.com> and regularly tweets as joelmonte

[Back To Index](#)




Teach-Testing →

An Ambitious Campaign by Tea-time with Testers



by





Call for Articles !

Have you got something to say?

yes, we are listening you...!!!

"Tea-time with Testers" firmly believes that one of the best ways to improve upon software testing is to listen to the lessons learned by others and their experiences too.

So, if you have an interesting story that you'd like to share with the world, contact us at teatimewithtesters@gmail.com.

Submit your articles, stories, thoughts around software testing.

now its your chance to be heard...!

Click [HERE](#) to read our Article Submission FAQs !

T ' Talks



T. Ashok exclusively on software testing

Do less work - 'Test case immunity' can help.

Test cases and the defects are central to what we do. We focus on uncovering more with the fervent hope of finding less later. We are also constantly challenged in doing this with less effort and time continually.

In my numerous interactions with test and management folks, the conversation always veers to two key questions - (1) Am I doing enough? Is my 'coverage' good enough? (2) How do I optimize and do it faster? I have noticed that most often test automation is touted as the solution for (1) & (2) - The ability to cover more area with less effort/cost using technology. I understand, but feel that this does not go far enough to achieve the real solution for (1) & (2).

This is when I started thinking deeply on the story "The pesticide paradox", particularly for (2). My line of thinking was "How can I ascertain that my software has become immune to some of the test cases and therefore not execute them?" It is not doing faster and cheaper, but really about 'not-doing'.

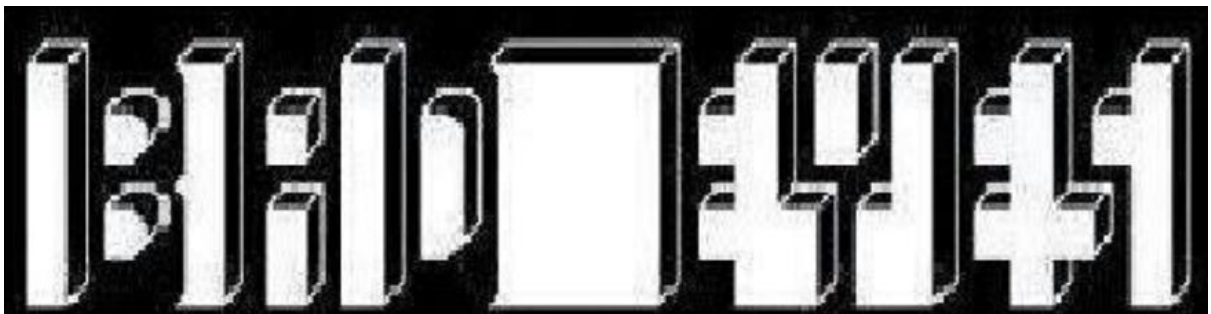
The story of pesticide paradox in brief...

"A poor farmer loses his crop and is advised to use pesticide. The next season around, he sprays the crop with DDT killing the pests and improving the yield. A few seasons later, the pests become resistant to DDT and now he is advised to switch to a different pesticide 'Malathion'. The yield improves but the story repeats again after a few seasons. This is the story of pesticide paradox, wonderfully illustrated by Dr Boris Bezier in his classic book "Software Testing Techniques".

The pesticide paradox is "The pest that you kill with a pesticide makes the pest resistant to that pesticide". This is used to illustrate the fact, that over time software too becomes 'resistant' to test cases i.e. test cases do not yield bugs.

Let's shift gears now... Let's look at defects and what we do with them. We use the defect data and produce reports that provide information about software quality and also about test quality. This is done by examining data related to defect rates, defect densities, defect distribution etc. i.e. we pay significant attention to defects. We know that as time progresses, the same test cases do not yield defects, then what do we analyze? Hmmm...

Look at the interesting picture below, What do you see? It depends on what you want to see and, at possibly on what distance you see it from. Focus on the black and move the eye farther from the picture and voila, you see a meaningful phrase instead of a mix of fat lines.



What am I getting to? If you chose to see defect information only, analyze them and use the information to make choices, then you are limited. On the contrary if you see "no-defect" (i.e. absence of defect) and at the same time shifting to a higher level view of seeing 'defect types'(rather the raw defects), you see new information suddenly, and this will help you find better answers for (2).

Setting up defect types (termed as Potential Defect Types- PDT in Hypothesis Based Testing 'HBT') and then categorizing the defects found into these types, and more importantly analyzing those defect types that have not surfaced (i.e. no defects of these types) allows us to understand as to which test cases do not have an yield.

If it can be proven that the test cases are indeed complete/adequate (in HBT this is done by assessing two properties of test cases - countability and fault traceability) then the absence of certain defect types

indicates "test case immunity" and thereof "hardening of software". This means that the area of the software being irritated by the test cases have hardened i.e. become immune and is clean. Hence focusing on this area of the software is therefore not logically useful and hence these test cases can be "parked". The net result is that we do less work and therefore achieve a higher degree of optimization.

I can visualize you shaking your head in disagreement and commenting "How can I 'park' these test cases not knowing if I may be leaving one 'minesweeper bomb' inside?"

OR

I do this anyway, as my experience enables me to figure out which test cases of the total I have to execute.

My take on this is: let us do this logically by examining the "categories of empty space" (i.e. absent defect types). In effect we are assessing the parts of the software system that have become immune to those types of defects that matter for those areas.

Remember that we are not examining the actual defects; rather we are examining the test cases that have passed, across the last few cycles of testing, with a clear knowledge of the type of the defect each test case is targeting on. This is examining the 'empty space'. It is however very necessary to ensure that the adequacy of test cases be logically proven before using this type of "test immunity analysis".

On a different note, we now know that empty space consists of dark matter that cannot be seen but probably shapes our universe. See the unseen. Enough of philosophy.

Mull over this. Every time a test case passes, don't pass over it, use this knowledge of "no defect" to logically analyze "immunity".

Do less work. May the force be with you.

Until next time CIAO.

[Back To Index](#)



T Ashok is the Founder & CEO of STAG Software Private Limited.

Passionate about excellence, his mission is to invent technologies to deliver "clean software".

He can be reached at ash@stagsoftware.com.





OUR PARTNERS

Quality Testing



Quality Testing is a leading social network and resource center for Software Testing Community in the world, since April 2008. QT provides a simple web platform which addresses all the necessities of today's Software Quality beginners, professionals, experts and a diversified portal powered by Forums, Blogs, Groups, Job Search, Videos, Events, News, and Photos.

Quality Testing also provides daily Polls and sample tests for certification exams, to make tester to think, practice and get appropriate aid.



Mobile QA Zone

Mobile QA Zone is a first professional Network exclusively for Mobile and Tablets apps testing.

Looking at the scope and future of mobile apps, Mobiles, Smartphones and even Tablets, Mobile QA Zone has been emerging as a Next generation software testing community for all QA Professionals. The community focuses on testing of mobile apps on Android, iPhone, RIM (Blackberry), BREW, Symbian and other mobile platforms.

On Mobile QA Zone you can share your knowledge via blog posts, Forums, Groups, Videos, Notes and so on.



Testing PUZZLES

by Sebi



Claim your **Smart Tester of The Month Award**. Send us an answer for the Puzzle and Crossword below by 15th July 2012 & grab your Title.

Send -> teatimewithtesters@gmail.com with Subject: Testing Puzzle

**Exciting
PRIZE for 1st
three
WINNERS***

NOTE : S.T.O.M. contest comprises of Testing Puzzle + Crossword. To claim their prize, participants should to send answers both for puzzle and crossword.

"Solve This Puzzle"



"Use exactly five '5's to form every integer from 0 to 55, using only the operators +, -, x, /, () (brackets) x^2 (square), and ! (factorial).

Example: $0 = (5-5)*555$ "



Biography



Blindu Eusebiu (a.k.a. Sebi) is a tester for more than 5 years. He is currently hosting European Weekend Testing.

He considers himself a context-driven follower and he is a fan of exploratory testing.

He tweets as @testalways.

You can find some interactive testing puzzles on his website www.testalways.com

Back To Index 



TESTING CROSSWORD



1		2		3		4	
5				6	7		
							8
9				10			
11							

Horizontal:

1. A software item which is the object of testing (8)
5. Running a system at high load for a prolonged period of time is called _____ testing (4)
6. A combination of Black Box and White Box testing is known as _____ Box Testing (4)
9. It is a utility for automating Java GUI tests (7)
11. Testing aimed at showing software does not work, it known as _____ testing (8)

Vertical:

1. It is an automated testing tool for testing interactive web systems (8)
2. It is the world leading Functional Testing Tool, mainly used for Web Service Testing (6)
3. A group of people whose primary responsibility is software testing, in short form (3)
4. Checks for memory leaks or other problems that may occur with prolonged execution is known as _____, in short form (2)
7. Continuously raising an input signal until the system breaks down, in short form (2)
8. A quick-and-dirty test that the major functions of a piece of software work is known as _____ testing (5)
10. Defect Removable Efficiency, in short form (3)

Answers for last month's Crossword:

T	E	S	T	D	A	T	A
E		E		E		E	
S	T	E	P	L		S	T
T		T		U	A	T	
C		E	G	G		U	T
A	T	S		E	B	F	
S		T				F	
E	M	U	L	A	T	O	R

Answer for last Puzzle:

If you put the letters in numbers, you get

112 117 122 122 108 101 32 105 115 32 115 111 108 118 101 100 33 32 99 111

and if you use ASCII to text converted you get "puzzle is solved! co"



We appreciate that you

"LIKE" US!



Join us on Facebook.

You are just a CLICK AWAY



Every Tester
who reads **Tea-time with Testers**,
Recommends it to friends and
colleagues .

What About You ?

Our Testimonials

“Tea-time with Testers” helps in exploring different aspects of software testing, thus enriching our knowledge. Keep continuing the good work!!

- Mahima Chourasia

I loved your magazine. It's really superb.

- Samantha Jose

I want this magazine every month, I love it !

Regards,
Kongdemin

in ne>xt issue

articles by -



IT'S
ALWAYS
TEA - TIME

Jerry Weinberg

T Ashok

Joel Montvelisky

Michael Bolton

Anurag Khode

Bernice Ruhland

Samarjeet Mohanti

our family

Founder & Editor:

Lalitkumar Bhamare (Mumbai, India)

Pratikkumar Patel (Mumbai, India)



Lalitkumar



Pratikkumar

Contribution and Guidance:

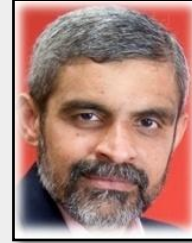
Jerry Weinberg (U.S.A.)

T Ashok (India)

Joel Montvelisky (Israel)



Jerry



T Ashok



Joel

Editorial | Magazine Design | Logo Design | Web Design:

Lalitkumar Bhamare

Image Credits- Indulgy / DK Photography

Core Team:

Anurag Khode (Nagpur, India)

Dr.Meeta Prakash (Bangalore, India)



Anurag



Dr. Meeta Prakash

Testing Puzzle & Online Collaboration:

Eusebiu Blindu (Brno , Czech Republic)

Shweta Daiv (Mumbai, India)



Eusebiu



Shweta

Tech -Team:

Chris Philip (Mumbai, India)

Romil Gupta (Pune, India)

Kiran kumar (Mumbai, India)



Kiran Kumar



Chris



Romil

*// Karmanye vadhikaraste ma phaleshu kadachina /
Karmaphalehtur bhurma te sangostvakarmani //*

To get **FREE** copy ,
Subscribe to our group at



Join our community on



Follow us on



www.teatimewithtesters.com

